НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 “ Комп’ютерні науки ”

на тему: «ГІС аналіз стану якості водних екосистем»

Виконав: студент IV курсу, групи ТМ-51

Кривоконь Єгор Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник старший викладач, Гурін А. Л. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “ Комп’ютерні науки”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)

”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту
Кривоконю Єгору Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи: «ГІС аналіз стану якості водних екосистем»
керівник роботи Гурін Артем Леонідович, старший викладач
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 201__р. № __

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи мова програмування Java, мова програмування JavaScript, середовище JetBrains IntelliJIDEA 2019, бібліотека GoogleChart.js_____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) ознайомитись з проведенням хімічного аналізу води, проаналізувати програми, що містять дані щодо хімічного аналізу води, розробити програмний засіб для дослідження та аналізу якості стану водних екосистем України

_____

5. Перелік ілюстративного матеріалу архітектура системи, графічне представлення інтерфейсу, приклади роботи програмного модулю_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	14.10.2018-23.12.2018	
2.	Вивчення та аналіз задачі	2.02.2019-3.03.2019	
3.	Розробка архітектури та загальної структури системи	4.03.2019-14.04.2019	
4.	Розробка структур окремих підсистем	15.04.2019-18.04.2019	
5.	Програмна реалізація системи	18.04.2019-14.05.2019	
6.	Оформлення пояснювальної записки	16.05.2019-3.06.2019	
7.	Захист програмного продукту	15.05.2019	
8.	Передзахист	28.05.19	
9.	Захист	17.06.2019-22.06.2019	

Студент _____ Кривоконь Є.О.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Гурін А.Л.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Дипломну роботу виконано на 58 аркушах, вона містить 3 додатки та перелік посилань на використані джерела з 15 найменувань. У роботі наведено 18 рисунків.

Метою даної дипломної роботи є створення програмного забезпечення для проведення ГІС аналізу стану якості водних екосистем України. Реалізацію програмного продукту було зроблено у вигляді веб-додатку для зручного та швидкого доступу користувачів з будь-яких операційних систем.

Ключові слова: ГІС аналіз, якість води, інтерактивна мапа, хімічний аналіз води, інтерактивна таблиця, водні ресурси.

ABSTRACT

The thesis is presented in 58 pages. It contains 3 appendixes and bibliography of 15 references. Eighteen figures are given in the thesis.

The goal of the thesis is to develop software tools for the GIS-analyze the status of the quality of water ecosystems in Ukraine. The implementation of the software was made as a web application for easy and fast access of users from any operating system.

Keywords: GIS analysis, water quality, interactive map, chemical analysis of water, interactive table, water resources.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД	База даних
SQL	Structured query language – мова структурованих запитів
СУБД	Система управління базами даних
MVC	Model-view-controller – модель-представлення-контролер
HTML	Hypertext markup language – мова розмітки гіпертекстових документів
DOM	Об’єктна модель документа
IDE	Integrated development environment – Інтегроване середовище розробки
CSS	Cascading style sheets – каскадна таблиця стилів
JSP	Java server pages – серверні сторінки Java
БСК	Біохімічне споживання кисню
ООП	Об’єктно-орієнтовне програмування
ГДК	Гранична допустима концентрація

ВСТУП

Вода – найбільш поширена неорганічна сполука на Землі. Хоча водні ресурси і належать до невичерпних природних ресурсів, але вони є невичерпними лише як фізичне тіло. Проте ці ресурсі підлягають значним негативним змінам в процесі діяльності людини.

Якість більше половини водних ресурсів України не відповідають нормативним вимогам. Це призводить до погіршення стану флори та фауни країни, а також є причиною зростання захворюваності людей. Чим швидше ми зможемо дізнатись про незадовільний стан води у різних точках країни і чим швидше вживатимемо заходів для відновлення цього природнього ресурсу, тим більший об’єм води може бути безпечно використано для задоволення потреб людини.

Внаслідок розвитку промисловості, транспорту, сільського господарства очищення води стає все більш важкою задачею, адже необхідно вчасно дізнатися місце погіршення стану якості води та його причину. Через це водні ресурси України деградують, а якість води значно знижується.

Забруднення викликає зміну характеру середовища й властивостей його компонентів, шкідливо впливає на розвиток живих організмів.

Ступінь змін і масштаби наслідків залежать від інтенсивності й характеру забруднення, від наявності у воді різних хімічних сполук та речовин, перевищення вмісту гранично допустимих норм яких призводить до погіршення стану екосистеми.

Саме тому необхідно мати можливість швидко дізнаватися актуальну інформацію про стан водних ресурсів на тій чи іншій території країни, адже це допоможе своєчасно почати вирішення проблеми забрудненості екосистеми, виявити її можливу причину та запобігти поширенню. Таку можливість було вирішено надати через створення веб-додатку. Адже таким чином усе, що необхідно

користувачеві для отримання інформації та проведення аналізу – це доступ то мережі «Інтернет».

Актуальність даної роботи викликана відсутністю програмного забезпечення, що надало би змогу швидко отримувати актуальну інформацію щодо стану якості водних екосистем України у будь який момент часу та можливість аналізу отриманих даних, при цьому з мінімальними вимогами до апаратного та програмного забезпечення.

Це допоможе своєчасно почати вирішення проблеми забрудненості екосистеми, виявити її можливу причину та запобігти поширенню.

Реалізацію програмного продукту було створено мовами Java (серверна частина) та JavaScript (клієнтська частина), адже поєднання таких систем розробки є оптимальним і сучасним підходом до розробки веб-додатків.

1 ПОСТАНОВКА ЗАДАЧІ

Метою даної дипломної роботи є реалізація програмного засобу, що надає можливість отримати інформацію щодо стану якості водних ресурсів України та провести ГІС аналіз з можливістю дослідити становище води за минулі роки та передбачити його зміну в майбутньому.

Для реалізації програмного продукту було обрано наступні технології:

1) Java 8 language
2) Spring MVC Framework
3) Maven
4) Hibernate Framework
5) HTML5
6) CSS3
7) JavaScript
8) JSP
9) MySQL

Необхідними можливостями, які має забезпечувати веб-додаток, є:

1) Інтерактивна мапа з відображенням стану водних екосистем;
2) Систематизація та перегляд інформації у вигляді діаграм і таблиць;
3) Збереження інформації на диск у вигляді Excel документу для доступу до даних офлайн;
4) Надання детальної схеми басейну основної річки обраної на мапі області;
5) Надання статичної мапи з річками, що складають основу водних ресурсів України;
6) Інтерактивна таблиця з даними про вміст у воді речовин та їх ГДК з можливістю табличного та графічного порівняння показників за останні три роки.

2 ЗАГАЛЬНИЙ АНАЛІЗ ПРАВИЛ ДОСЛІДЖЕННЯ ЯКОСТІ ВОДИ

Згідно з офіційними статистичними даними, стан більше 50% водних ресурсів України не відповідає нормативним вимогам. Низьку якість води часто буває видно навіть без спеціального обладнання та детального дослідження. Проте після проведення якісного хімічного аналізу, можна зробити багато висновків про причини забруднення та знайти найоптимальніший спосіб стабілізації ситуації [1].

2.1 Цілі лабораторного дослідження води

Такі параметри, як каламутність, велика кількість осаду, неприємний смак можна визначити самостійно, проте ці показники не є гарантією точного виявлення зниженням якості водного еко-ресурсу.

Головним завданням детального хімічного дослідження води є: оцінка принципової можливості використання джерела водозабору для питних і господарських потреб. Таким чином буде виявлено чи становить використання води в тій чи іншій області загрозу для здоров’я людини [2].

2.2 Необхідна частота проведення дослідження якості води

Частота проведення якісного хімічного дослідження залежить від віку і типу ресурсу, що досліджується. Дослідження не потребують ресурси, що

використовуються для міського водопостачання, тому що їх аналіз зобов’язані проводити відповідні комунальні служби.

Для усіх інших водних ресурсів проведення аналізу складу води бажане якомога частіше, проте нормою вважається проведення повного хімічного аналізу один раз на рік.

2.3 Правила проведення хімічного аналізу джерел

В ході хімічного аналізу досліджується:

1) органолептичні показники – колір, запах, каламутність
2) компонентний склад – фізичні та хімічні властивості

Після проводження аналізу, усі отримані результати звіряються з встановленими ДСанПіН 2.2.4-171-10 нормами. Цей документ визначає межу гранично допустимої концентрації (ГДК) вмісту у воді певних хімічних речовин. В залежності від того, наскільки перевищено ГДК вода може бути придатною, малопридатною або непридатною до використання.

Основні показники ГДК:

1) ГДК заліза не повинна перевищувати 0,3 мг/л,
2) хлоридів - 350,
3) нітратів - 45, марганцю - 0.1,
4) алюмінію - 0.5 м /л,
5) жорсткість не повинна бути вище 7 мг-екв /л,
6) водневий показник - бути в межах 6 ÷ 9 од. РН.

Базова методика передбачає оцінку 10-15 показників [3]. В рамках дипломної роботи для проведення якості аналізу було взято наступні речовини:

1) Нітрит. іони

2) Поліфосфати
3) Синтетичні поверхнкві активні речовини
4) Хімічне споживання кисню
5) БСК5
6) Завислі супендовані речовини
7) Кисень розчинений
8) Сульфат. іони
9) Хлорид. іони
10) Амоній. іони
11) Нітрат. іони

Усі дані є актуальними, адже взяті з офіційного джерела: державного агентства водних ресурсів.

2.4 Висновки до розділу

В даному розділі було проаналізовано та вивчено правила проведення хімічного аналізу води, результати якого буде використано при розробці веб-додатка. Було визначено речовини, які враховуватимуться при проведенні аналізу стану якості води.

3 АНАЛІЗ ІСНУЮЧИХ ПРОГРАМ АНАЛОГІВ

Існує декілька веб-додатків за функціоналом подібні до розробленого. Вивчення вже існуючих програм для дослідження стану якості води дозволило зробити корисні висновки, які було враховано при створенні власного веб-додатку.

3.1 Веб-ресурс «WaterNet»

Ресурс «WaterNet» (рисунок 3.1) має досить зручний інтерфейс з інтерактивною мапою, якою легко користуватись. Також даний веб-додаток надає дані у зручному, табличному вигляді.

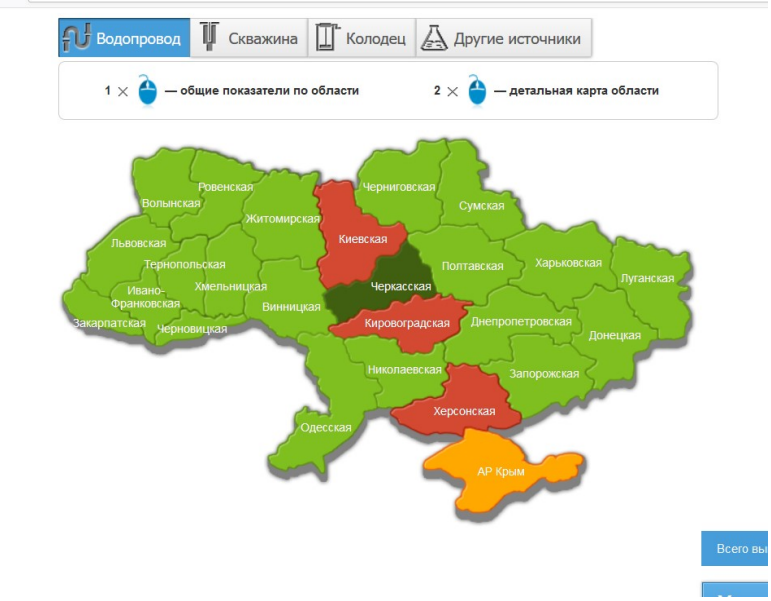

Рисунок 3.1 – Інтерфейс ресурсу «WaterNet»

Основним недоліком даного ресурсу є відсутність статистичних даних за попередні роки, таким чином є неможливим проведення аналізу та виявлення закономірностей у забрудненні водного середовища. Також не вказано джерело, з якого було взято представлені дані.

3.2 Ресурс «Чиста вода»

Основою ресурсу «Чиста вода» (рисунок 3.2) є зручна інтерактивна мапа з даними щодо стану якості води у багатьох місцях України. Дані представлені офіційні та за декілька останніх років, що також є перевагою даного веб-сервісу.

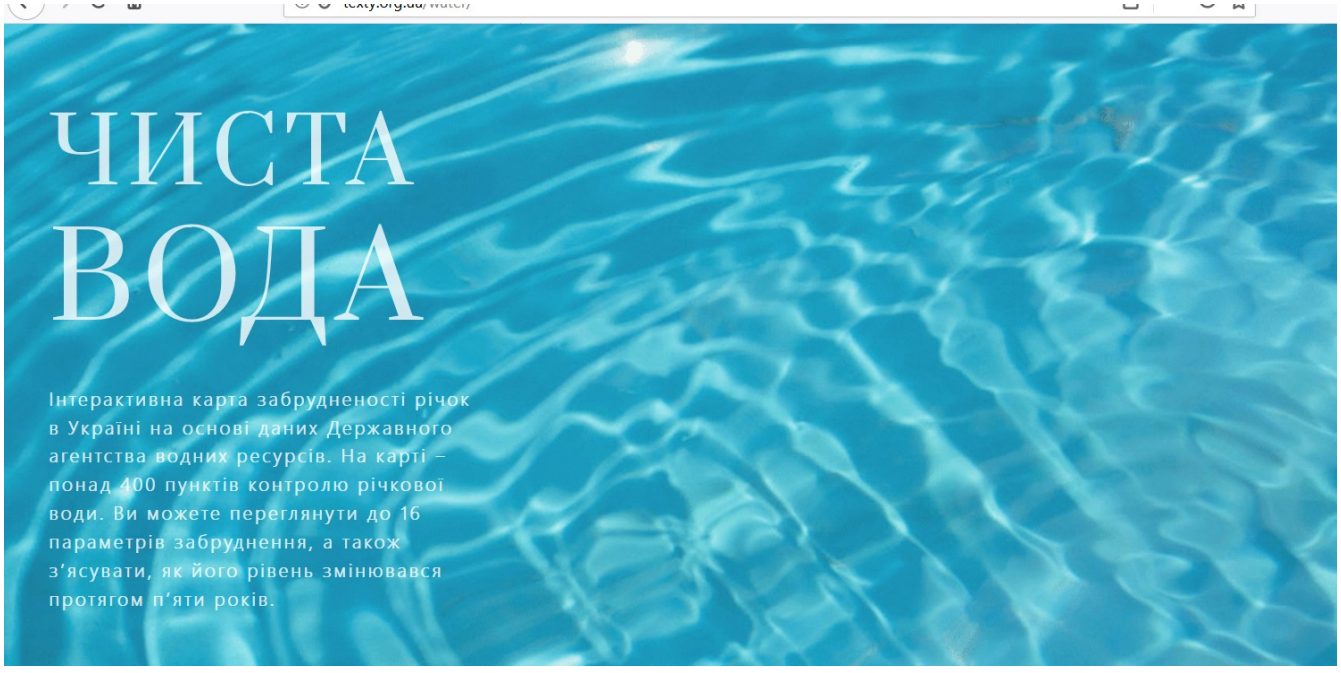

Рисунок 3.2 – Інтерфейс ресурсу «Чиста вода»

Проте в цьому веб-додатку немає жодної можливості для збереження даних у зручному для користувача форматі для доступу до них без мережі інтернет.

Також через велику кількість анімації та велику кількість зображень сайт працює досить повільно та може бути недоступним при повільній швидкості інтернету.

3.3 Ресурс «Державне агенство водних ресурсів»

Державне агенство водних ресурсів є офіційним джерелом, а отже, містить лише актуальну інформацію. Також цей веб-севріс дає змогу зберігати дані за декілька останніх років на диск в форматі PDF. Таким чином, ми можемо переглядати інформацію з ресурсу навіть не маючи доступу до нього. Інтерфейс даного ресурсу зображено на рисунку 3.3.

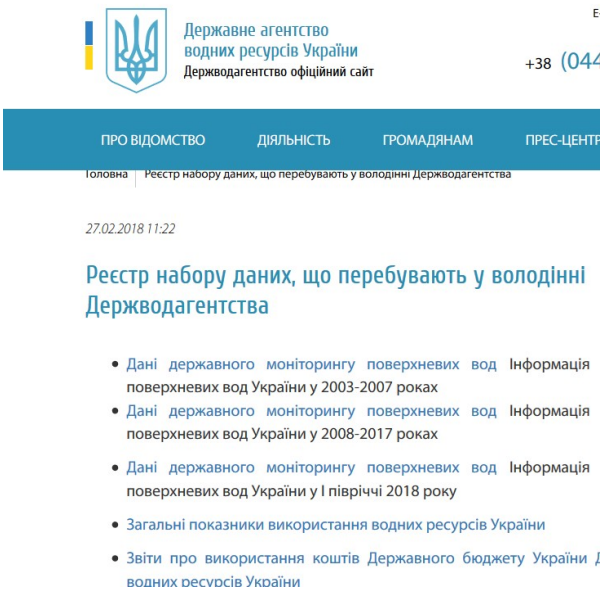

Рисунок 3.3 – Інтерфейс ресурсу «Державне агенство водних ресурсів»

Але навідіну від перелічених вище ресурсів, даний веб-додаток не містить майже ніякої візуалізації інформації. Це робить неможливим зручний, детальний аналіз даних про стан якості водних екосистем.

3.4 Висновки до розділу

В даному розділі було проведено аналіз існуючих програм аналогів, виявлено їх недоліки та переваги, які буде враховано при створенні власного програмного продукту. Веб-додаток має надавати користувачеві лише актуальну й офіційну інформацію, також усі наведені дані повинні бути структуровані та надані у формі графіків, діаграм та таблиць для можливості проведення користувачем їх аналізу.

Також має бути надана можливість зберегти дані в Excel форматы для доступу до них за відсутності доступу до мережі «Інтернет».

4 ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОГРАМИ

Важливим аспектом розробки програмного продукту є правильний вибір стеку технологій, що будуть використані. Адже лише при правильному, оптимальному поєднанні мов програмування, бібліотек, бази даних та інших засобів розробки можна досягти правильного функціонування системи.

Під час створення програмного продукту було використано програмні засоби, зображені на рисунку 4.1.

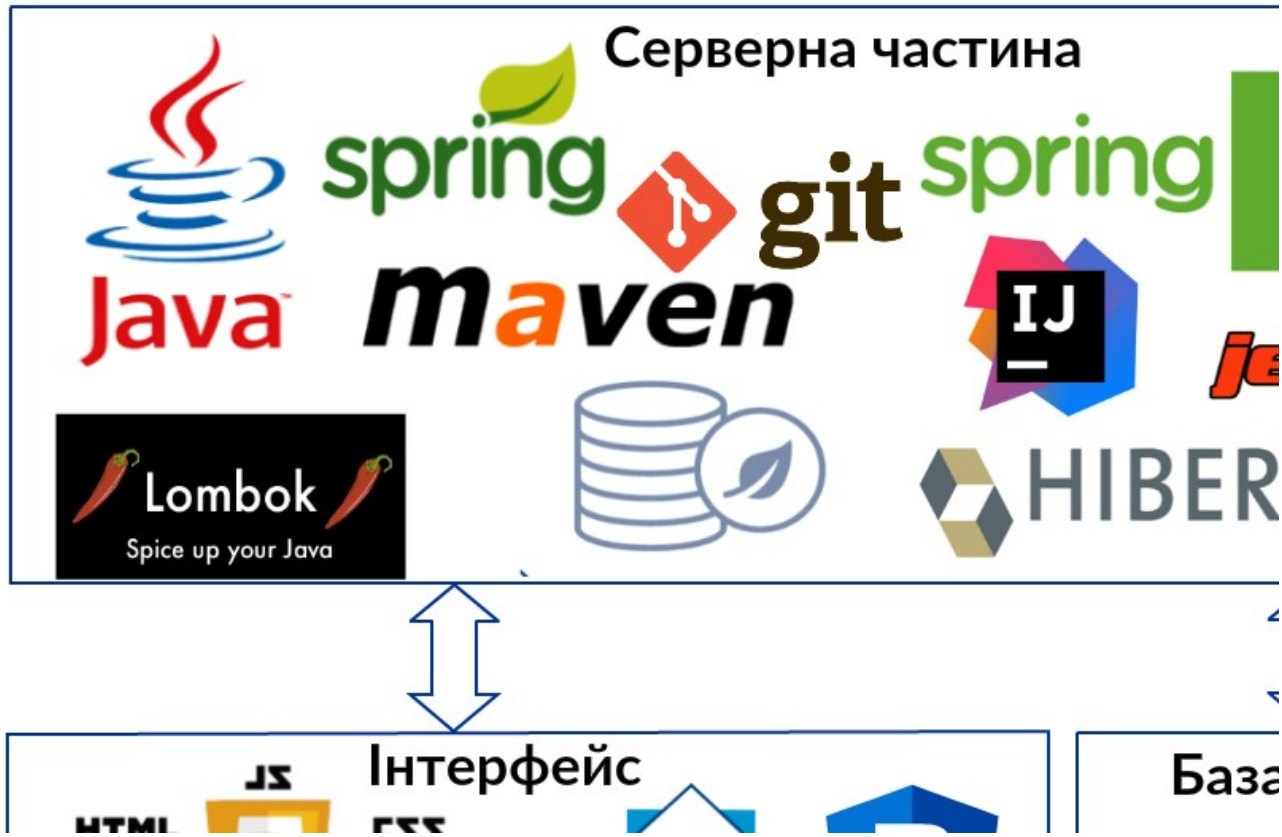

Рисунок 4.1 – Використані засоби реалізації

— Середовище розробки IntelijIDEA, що має дуже зручний інтерфейс та надає багато можливостей при розробці програм;

— Мова програмування Java для створення back-end частини;

— MVC Framework для організації архітектури додатку;

— Шаблон проектування архітектури додатку Model-View-Controller для поділу моделі і її представлення, що необхідно для їх зміни окремо один від одного;

— Hibernate для доступу до даних бази даних;

— Spring для реалізації MVC;

— База даних MySQL.

Розробка саме веб-додатку обумовлена тим, що таким чином усе, що необхідно для використання програми це доступ до інтернету і браузер.

Основною мовою програмування була обрана Java, адже вона не лише кросплатформенна, але й має багато бібліотек, що допомагають спростити процес розробки програми, роблять архітектуру проекту більш чіткою і зрозумілою, що є перевагою під час розширення проекту.

4.1 Мова програмування Java

Логіка серверної частини програми була реалізована на мові Java. Ця мова програмування високо рівня є об’єктно орієнтовною. Для Java розроблено багато бібліотек, які вирішують спільні для більшості проектів проблеми. Їх використання значно пришвидшує процес розробки. Також Java є кроссплотформною мовою, тому користуватись веб-додатком можна з будь-якої операційної системи.

Так як Java розроблялась як об'єктно-орієнтовна мова і має жорстку типізацію, то надійність роботи, як і кроссплатформність, також можна віднести до її основних переваг. Прикладом оптимізації обраної мови є також і виключення з неї множинного наслідування яке часто призводить до неоднозначності у коді. Натомість розробникам було надано такий засіб як інтерфейс, за допомогою якого можна безпечно відтворити логіку, подібну для множинного наслідування. Ще

одним показником надійності мови є гарно продумана система помилок та виняткових ситуацій.

У Java існує два типи помилок:

— ситуації, коли помилки виникають до запуску програми, проте не суперечать синтаксису і проявляються лише після компіляції і запуску. Такі виняткові ситуації мають передбачатися та оброблятися до запуску програми, наприклад: програмний продукт намагається отримати доступ до файлу, якого не існує на диску, або користувач вводить символи в поля для вводу цифр;

— ситуації, коли помилку неможливо виявити до запуску та роботи з програмою. Прикладом такого виключення є переповнення пам’яті, що призводить до помилки переповнення стеку [4].

Для того щоб полегшити контроль над використовуваною програмою пам’яттю у Java було добавлено збирач сміття. Так, якщо наприклад у мові C++ програміст має сам створювати об’єкт, а після закінчення роботи з ним видаляти його, то в Java процес видалення об’єкту з пам’яті є автоматичним. Об’єкт буде видалений збирачем сміття, якщо на нього більше не вказує жодне посилання. Таким чином, обрана мова програмування автоматично вміє керувати пам’яттю, що є суттєво для оптимізації програми.

На відміну від таких мов, як C та C++, Java не підтримує роботу з вказівниками. Таким чином, збирач сміття має більше контролю над вказівниками на об’єкти та самостійно переміщує їх на необхідну ділянку пам’яті.

Зручним є і об’єктно-орієнтовний підхід до написання програм, адже для нього легко і швидко розробляти архітектурні рішення, оскільки є паралелі з реальним світом [5]. Завдяки ООП підходу, різні модулі програми можна використовувати багато разів, що допомагає уникнути повторюваності коду. Також функціонал, написаний у такому стилі легко розшири і підтримувати. Це називається масштабованістю і є ще однією перевагою ООП підходу.

Оскільки програми, написані на мові Java, можуть бути запущені на будь якій системі де встановлено JDK, ця мова широко використовується та є дуже популярною у наш час.

4.2 Spring фреймворк

Для того, щоб полегшити створення архітектури проекту та використання шаблонів проектування, часто використовуються спеціально розроблені для цього бібліотеки – фреймворки.

Software framework (програмний фреймворк) — набір розроблених готових рішень, що може бути використаний при розробці програмного продукту та містити готові модулі починаючи від архітектури та логіки і закінчуючи дизайном.

Фреймворки містять у собі готові рішення, що можуть бути використані як для полегшення роботи з середніми та великими проектами, так і швидко створити структурований невеликий проект. Також фреймворк може містити різноманітні скрипти, навіть готові програмні модулі. Оскільки кожен фреймворк має чітку структуру проекту, то дуже просто читати та змінювати код написаний іншими програмістами на одному фреймоврку.

Spring — фреймворк, створений для пришвидшення і розробки веб-додатків, створенням яких може займатись як один програміст, так і ціла команда. Spring має багато готових модулів та бізнес-рішень. Код даного фреймворку є відкритим і вільно розповсюджується.

Spring повністю контролює структуру додатку, таким чином дозволяючи програмісту не хвилюватись про архітектуру продукту, а займатись реалізацією бізнес логіки. [6]. Spring дозволяє розробляти усі види додатків на мові Java і однаково гарно підходить для створення як веб-додатків, так і JEE додатків, що

позитивно відрізняє Spring від багатьох інших платформ, які, зазвичай, є спеціалізованими та вузько направленими, що значно зменшує їх популярність та актуальність. [7].

У Spring дотримано наступних принципів розробки для спрощення і підтримання консинстентності коду:

— слабке зв’язування, що досягається за допомогою ін’єкції залежностей в конструкторі класу, або сетери, якщо поля класу є приватними. Інтерфейси також є засобом досягнення слабкого зв’язування адже кожен клас може мату свою, унікальну реалізацію інтерфейсу незалежно від інших його наслідників;

— інверсія управління, яка допомагає створювати незалежні один від одного модулі програми та паралельно розвивати їх. Особливо корисним це є при командній розробці;

— декларативне програмування за певними прийнятими угодами, завдяки чому код зчитується та розуміється досить легко;

— використання Plain Old Java Object, тобто простих Java об’єктів, при цьому програміст не зобов’язаний реалізовувати вбудовані у фреймворк інтерфейси чи абстрактні класи;

— Spring керує життєвим циклом об’єктів, що знаходяться у ньому.

Ін’єкция залежностей (англ. Dependency Injection) — це паттерн проектування, який використовується для того, щоб відтворити функціонал певного об’єкта через інший об’єкт. Так, загальний механізм займається створенням необхідних іншим об’єктам залежностей. Таким чином ми маємо можливість відділити роботу та реалізацію конкретного об’єкта від реалізації об’єктів, які він використовує як залежності отримавши завдяки цьому гнучкість при розробці продукту. Прикладом використання ін’єкції залежності є передача у конструктор строки з’єднання з базою даних, замість того щоб з’єднуватися з БД в конструкторі, прив’язуючись таким чином до конкретної реалізації.

Основним модулем Spring фреймворка є Core Container, що в свою чергу складається з підмодулей: Beans, Context, Core, SpEL. З Core модулем взаємодіють всі інші складові фреймворка [8].

Spring Framework має широкий стек технологій (рисунок 4.2)

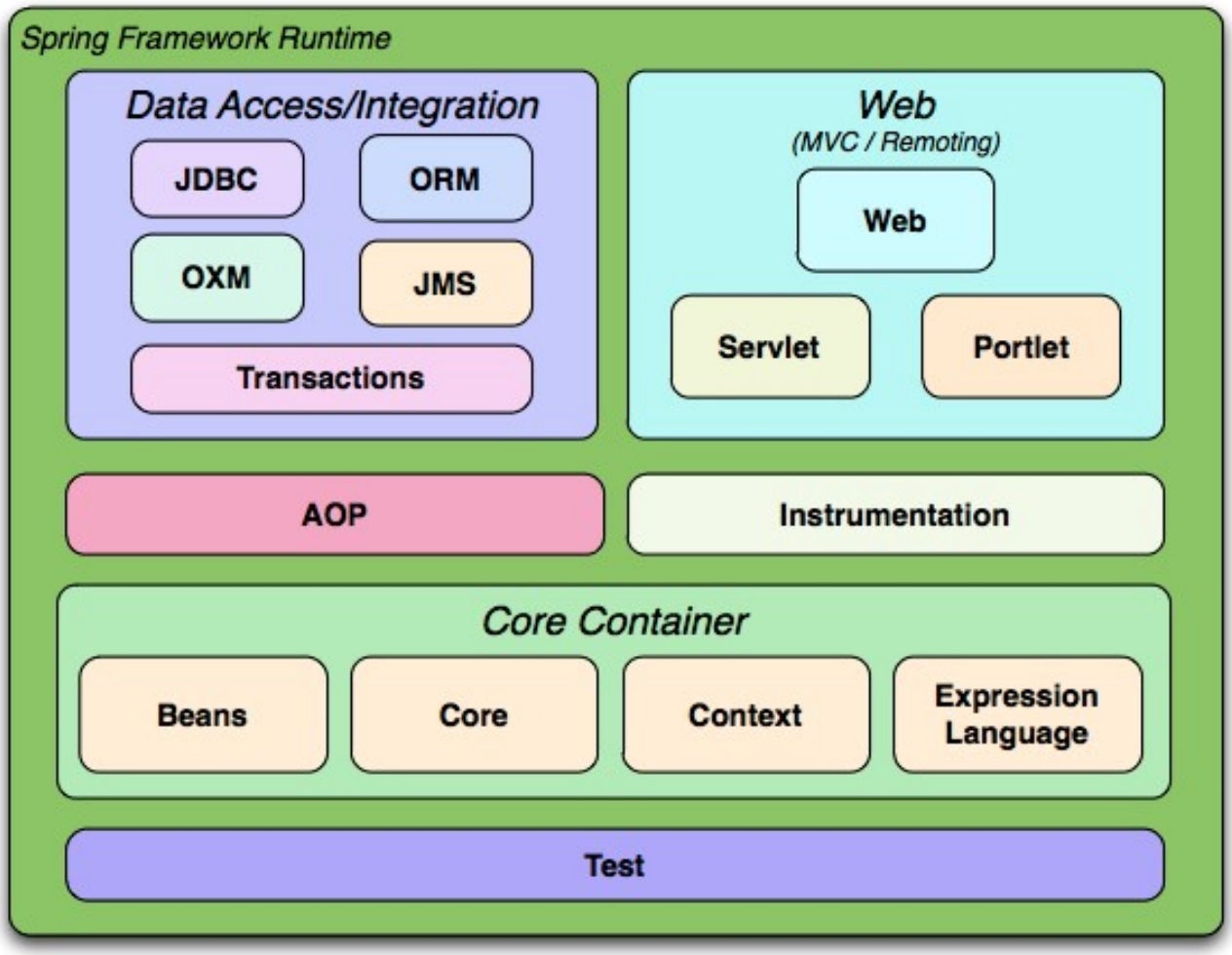

Рисунок 4.2 — Стек технологій Spring Framework

Beans та Core заюезпечують найбільш важливі частини фреймворку — інверсію управління та ін’єкцію залежностей, забезпечують реалізацію шаблона фабрики, що усуває необхідність реалізовувати паттерн Singleton та відокремлює конфігурацію з специфікацією від фактичної логічки програми.

За допомогою Context модуля здійснюється доступ до об’єктів та реалізовується інтернаціоналізація.

SpEL (англ. Spring Expression Language) додає підтримку мов запитів, а саме: доступ до контексту масивів, встановлення і отримання значень властивостей, зчитування з конфігураційного файлу, колекцій та індексаторів, логічних і арифметичних операцій, пошук об'єктів по імені, виклик методу.

Модуль Data Access/Integration слугує для реалізації роботи з базою даних. Так, підмодуль ORM (англ. Object-relational mapping) слугує для вирішення задач об’єктно-реляційного відображення засобами JPA, JDO, Hibernate та iBatus.

Підмодуль JDBC (Java DataBase Connectivity) слугує для утворення взаємодії Java застосунку з різними СУБД.

Підмодуль Transaction реалізує створення та підтримку транзакцій.

Модуль AOP додає підтримку аспектно-орієнтованого програмування для більш гнучкого застосування додатку.

Модуль Instrumentation слугує для контролювання та покращення оптимізації і загальної швидкості роботи програми.

Модуль Test слугує для написання Unit та Integration тестів.

Модуль Web слугує для поєднання бізнес логіки з інтерфейсом користувача.

Компонент Web-Servlet слугує для можливості реалізації шаблону MVC (англ. Model View Controller — модель-представлення-контроллер) а також HTTP протоколу для веб-застосунку. За допомогою протоколу HTTP користувач та сервер передають дані один одному за допомогою запитів. На рисунку 3.3 зображено схему роботи шаблону. Згідно з ним забезпечується чітке розділення доменної (англ. domain) моделі, у якій міститься бізнес логіка, від веб форм, які слугують для відображення інформації користувачеві.

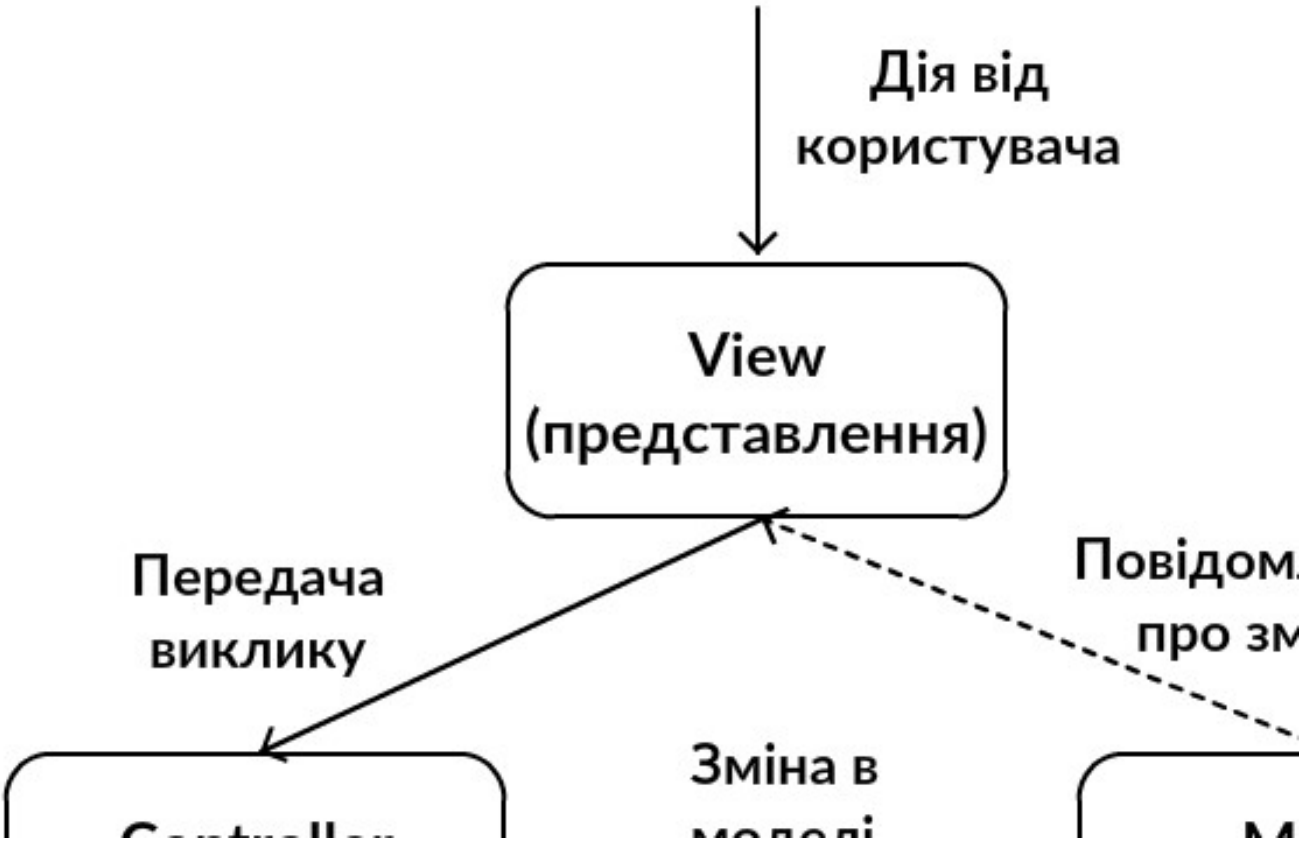

Рисунок 4.3 — Шаблон проектування MVC

Web Socket – завдяки цьому модулю між клієнтом та сервером підтримується двостороння комунікація.

В Spring Web MVC є можливість використовувати будь-який об'єкт в якості команди або об'єкта зі зворотним зв'язком; відсутня необхідність реалізовувати будь-якої спеціальний інтерфейс фреймворка або базовий чи абстрактний клас. Завдяки гнучкості зв'язування даних в Spring, невідповідність типів розглядається як помилки валідації і тому це може бути оброблено в додатку, а не в якості системних помилок.

Таким чином, немає потреби дублювати властивості бізнес-об'єктів, як простих нетипізованих рядків для ваших об'єктів форм. Тому можна легко обробляти неправильні підтвердження або правильно конвертувати їх в рядки. Замість цього, бажано пов'язувати такі об'єкти безпосередньо з об'єктами бізнес логіки.

Даний модуль Spring працює на основі центрального сервлету, який називається Dispatcher Servlet, завданням якого є розподілення запитів між контролерами. В ньому налаштовується обробка запитів, локалізація та часові зони [9]. Dsipatcher Servlet (з англ. Диспетчер сервлетів), це звичайний сервлет, що реалізує протокол HTTP. На рисунку 3.4 показана схема роботи цього сервлету з програмою.

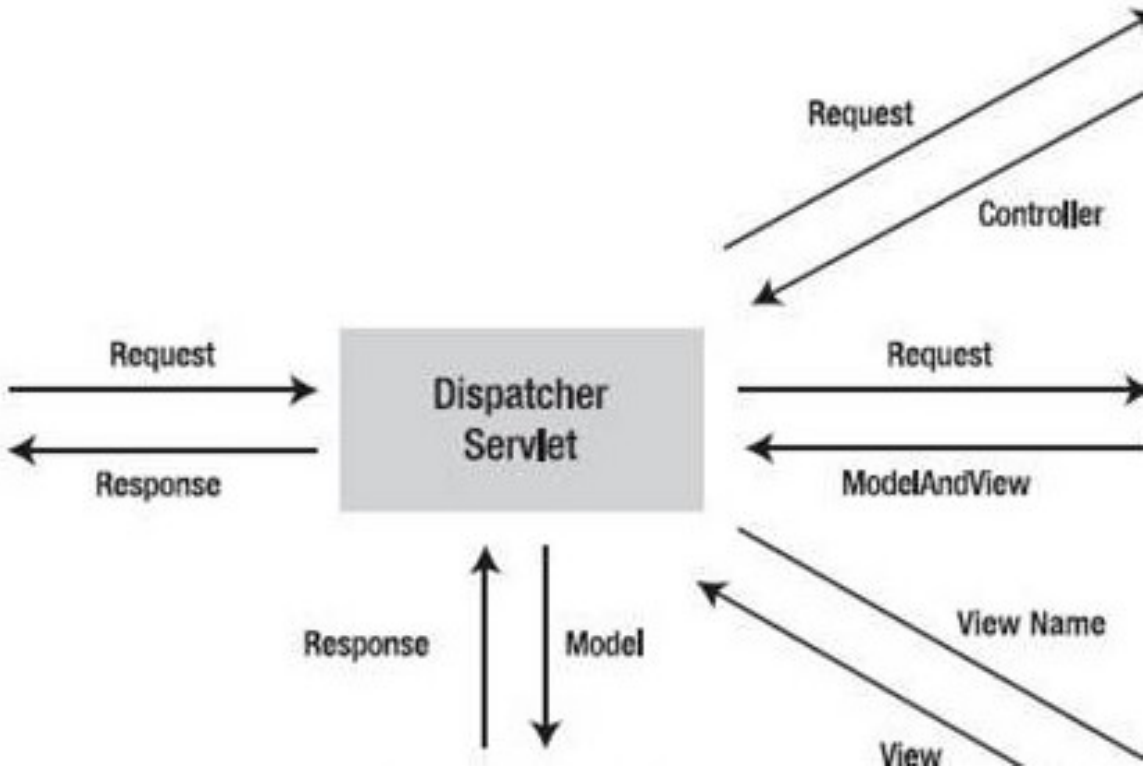

Рисунок 4.4 — Схема роботи Dispatcher Servlet

Спочатку запит потрапляє до Dispatcher Servlet, після чого він передає його у Handler Mapping, що переглядає усі наявні контролери та знаходить той, який знає як обробляти цей запит та повертає його ім’я назад до Dispatcher Servlet.

Ситуація зі знайденими двома контролерами, що мають код для обробки запиту неможлива, адже про неї розробнику буде повідомлено ще до запуску

проекту і без внесення необхідних змін, проект не запуститься. Після отримання імені контролера запит передається у нього. В контролері відбувається обробка запиту та назад відправляється об’єкт ModelAndView, що містить в собі дані та як їх потрібно відображати. Dispatcher Servlet на основі цього об’єкту шукає відповідне представлення (ім’я об’єкт типу View) за допомогою View Resolver. Завдяки цьому диспетчер сервлетів знає у яке представлення відправити модель (об’єкт типу Model), що містить дані. Від модуля представлення диспетчер сервлетів отримує відповідь, яка потім відсилається на веб-частину застосунку, у випадку коли вона необхідна.

Spring організовує архітектуру проекту шляхом зобов’язування програміста вказувати через анотації до якого типу класів відноситься даний, якщо ж не вказати тип, то фреймворк не буде поміщати даний клас у свій контейнер, а отже в ньому не будуть працювати методи з бібліотек, що надає Spring. Існує 4 анотації, що утворюють архітектуру проекту:

— @Component позначається приналежність класу до фреймворку;

— @Controller позначається клас, в якому Dispatcher Servlet буде шукати метод, що оброблює запит;

— @Service позначається клас, в якому реалізується бізнес логіка, нічим кардинально не відрізняється від @Component, але дозволяє розробнику вказати смислове навантаження класу;

— @Repository позначається клас, що буде використовуватися для роботи з пошуком, отриманням та зберіганням даних. Здебільшого дані класи використовуються для організації взаємодії з базою даних та реалізації відповідних шаблонів.

У результаті використання даного фреймворку отримаємо архітектуру проекту як показано на рисунку 4.5.

Рисунок 4.5 — Архітектура проекту з використання Spring фреймворку

4.3 Фреймворк Hibernate

Hibernate слугує для об’єктно-реляційного відображення Java класів та має відкритий код. Він надає можливість для відображення об'єктно-орієнтованої моделі даних в традиційні реляційні бази даних [10]. Hibernate не тільки забезпечує відображення Java класів в таблиці БД (і типів даних Java в типи даних SQL), але також забезпечує запит даних і пошукові засоби, що значно скорочує час розробки який витрачається на написання SQL і JDBC коду вручну.

Відповідність Java-класів таблицям з БД може бути досягнуто двома способами: за допомогою XML конфігурацій, за допомогою Java аннотацій. Hibernate дозволяє реалізовувати між класами зв’язок один-до-багатьох та багато-

до-багатьох. В доповення до управління асоціацію між об'єктами, Hibernate може також забезпечує рефлексивні відносинами, де класи мають зв'язок один-до-багатьох з іншими екземплярами свого типу.

Цей фреймворк не зобов’язує реалізовувати ніякі інтерфейси для своєї роботи, вся приналежність до фреймворку організовується через анотації, що вказують Hibernate як їх використовувати при зіставленні з базою даних. Під час виконання програми він зчитує ці анотації і використовує цю інформацію для того, щоб добудувати запити для відправки в деяку реляційну БД.

Hibernate реалізує інтерфейс JPA (Java Persistence API) — специфікація API Java EE, за допомогою якої можна зберігати об’єкти. Так, згідно неї, сутності — POJO-класи зв’язуються з БД за допомогою анотації @Entity, або через xml файл. POJO клас повинен відповідати таким умовам:

— мати пустий конструктор;

— не може бути вкладеним, інтерфейсом чи перерахуванням;

— не може бути з константою та мати final поля;

— мати хоча б одне поле помічене анотацією @Id.

Даний фреймворк складається з таких компонентів:

— EntityManagerFactory — фабрика EntityManager, що створює екземпляри диспетчера об'єктів, всі екземпляри виконані з можливістю підключення до тієї самої бази даних, щоб використовувати одні і ті ж параметри за замовчуванням, як це визначено в конкретної реалізації. Є можливість підготувати кілька фабрик entity manager отримати доступ до кількох сховищ даних;

— EntityManager використовується для доступу до бази даних в конкретній одиниці роботи. Він використовується для створення та видалення стійких сутностей, щоб знайти об'єкти по їх первинному ключу ідентичності, і для запиту по всім організаціям;

— Persistence context являє собою набір екземплярів об'єкта, в якому для будь-якої постійної ідентичності об'єкта є унікальний екземпляр об'єкта. У

persistence context, екземпляри сутностей і їх життєвий цикл управляється конкретним менеджером сутностей. Обсяг цього контексту може бути або транзакція, або розширена одиниця роботи.

4.4 Фреймворк Apache Maven

Apache Maven — це фреймворк, за допомогою якого можна швидко зібрати проект, описавши всі необхідні для нього залежні бібліотеки POM (англ. Plain Object Model) файлах. Відмінністю Maven від інших засобів збирання проектів є використання не імперативної, а декларативної збірки програми.

У файлах для опису проекту немає команд для збирання, натомість вони містять специфікацію для збірки. За допомогою системи різних плагінів, Maven обробляє і виконує завдання по обробці файлів, згідно з описаною специфікацією.

Стандартна структура папок проекту – обов'язкова вимога при користуванні Maven. Це є плюсом, коли над проектом працює багато розробників. Так, у кореневому каталогу має знаходитись лише папка src, в якій знаходиться програмний код; папка target, яка зберігає файли, створені у процесі роботи, Maven файли та pom.xml файл у якому вказуються залежності, плагіни та інші специфікації для роботи Maven з проектом. В директорії src містяться ще дві папки — main, в якій зберігається програмна реалізація продукту та test — яка зберігає Unit тести.

Maven керує життєвим циклом проекту. Для визначення порядку дій при побудові проекту Maven використовує певні фази [11]. Порядок виконання Maven команд при збиранні проекту:

clean — життєвий цикл для очищення проекту;

default — основний життєвий цикл, що включає в себе такі фази як:

validate, що виконує перевірку, чи є структура повною та правильною;

compile, що компілює код програми;

test, що запускає усі тести за папки test;

install — установка програмного забезпечення в локальний Maven-репозиторій, щоб зробити його доступним для інших проектів поточного користувача.

deploy — стабільна версія програмного забезпечення поширюється на віддалений Maven-репозиторій, щоб зробити його доступним для інших користувачів.

site — життєвий цикл генерації проектної документації.

4.5 Висновки до розділу

В розділі було перераховано технології, які буде використано при створенні програмного продукту. Кожна технологія була описана та вибір її обгрунтований. Таким чином, веб-додаток буде створено сучасними засобами програмування, що значно полегшить його розробку, розвиток і підтримку.

5 МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для надання можливості проведення аналізу якості стану водних екосистем та отримання необхідної інформації потрібно надати користувачеві швидкий веб-сервіс зі зручним інтерфейсом, який дозволяв би швидко і легко користуватись всіма можливостями сервісу.

Рішення реалізувати продукт як веб-сервіс було обумовлено тим, що користувач повинен мати змогу швидкого доступу до ресурсу, не встановлюючи при цьому жодного програмного забезпечення на своєму комп’ютері. Таким чином сервіс буде невимогливим до програмного забезпечення та системних вимог з боку користувача.

Вибір бази даних MySQL обумовлений простотою її використання, швидкістю роботи та можливість інтегрування з нею більшості сучасних мов програмування та засобів розробки програмного забезпечення.

Для організації архітектури програмного продукту було вирішено використати мову Java та JavaScript для візуалізації, адже такий підхід дозволяє розробити гнучкий програмний продукт з чітко виділеними шарами, реалізацію яких можна змінювати без зміни функціональності застосунку.

5.1 Системні вимоги

Розроблений веб-сервіс є невимогливим до системних чи апаратних вимог. Все, що необхідно користувачеві для користування програмою – це доступ до мережі «Інтернет» та веб-браузер. Мінімальні вимоги, необхідні для роботи з програмою наведені в таблиці 5.1

Пристрій	Характеристика
Процесор	З тактовою частотою не нижче 1.5 GHz: • Intel ® Core ™ 2 • 2 Duo • Pentium ® • Celeron ® • Xeon™ / i3 / i5 / i7 чи AMD 6 • Turion ™ • Athlon ™ • Duron ™ • Sempron ™
Оперативна пам’ять (RAM − Random Access Memory)	Рекомендовано не менше 1 RAM
Швидкість з’єднання з інтернет	Рекомендовано не менше 128 кб/сек

Таблиця 5.1 - Вимоги до апаратного забезпечення клієнта

Підтримувані апаратні архітектури:

- 32-розрядна (x86);
- 64-розрядна (x64)

Програму було протестовано з представленими у таблиці конфігураціями і не було виявлено жодних проблем з функціональністю системи.

5.2 Структура програми

Розроблена система представляє собою кросбраузерний та кросплатформенний веб-застосунок, взаємодію модулів якого показано на рисунку 5.1. взаємодія клієнта з сервером відбувається на основі HTTP протоколу. Модуль, що приймає запити реалізований фреймворком Spring MVC.

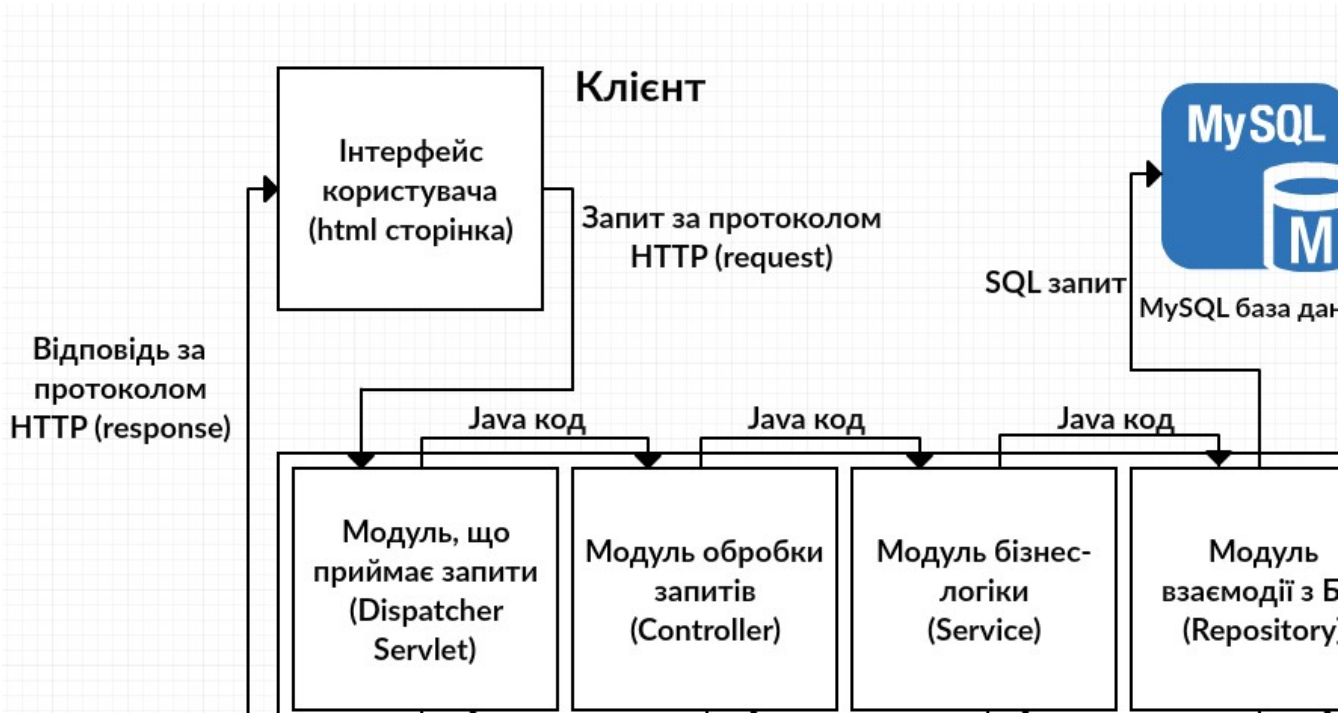

Рисунок 5.1 – Схема взаємодії модулів програми

Програмний продукт було реалізовано з використанням Spring фреймворку, це означає, що він має чітко розшаровану архітектуру, де кожен шар може бути замінений.

Структура проекту складається з наступних модулей: src, target, .idea, External Libraries та файлів .gitignore, та pom.xml.

Модуль src поділяється на main та test, а main у свою чергу на java, resources та webapp. Модуль src містить усю логіку застосунку; target містить скомпільований проект, який потім розгортається плагіном на сервері; .idea — це згенерований програмою середовища розробки модуль, що містить інформацію про проект [12]; External Libraries — це модуль, що містить набір підключених сторонніх бібліотек; файл з розширенням .iml створений для зручнішого імпорту проекту у інші проекти; файл .gitignore — містить список файлів, які не потрібно версіонувати та відправляти до віддаленого репозиторію, зазвичай це згенеровані середовищем розробки файли та файли що генеруються під час виконання програми; pom.xml — це основний файл проекту, що використовує Maven фреймворк, у ньому знаходиться основна інформація про проект, підключаються бібліотеки, підключаються плагіни необхідні та визначається порядок компіляції модулів.

Як вже зазначалося раніше, модуль src містить main, який поділяється ще на три модулі:

— папка java містить код серверної частини, який написано на мові програмування Java, а також конфігураційні організації роботи з базою даних та взаємодії з веб-інтерфейсом користувача;

— папка resources містить файли у яких лежить інформація для підключення бази даних та конфігураційних файл для роботи Hibernate фреймворку;

— папка webapp містить код реалізації інтерфейсу користувача, а також буфер у яких завантажуються файли на сервер від користувача.

Відповідно до архітектури Spring MVC фреймворку сервера частина коду програмного застосунку поділяється на наступні шари: controller, service та domain; Оскільки, для конфігурації було вирішено використати підхід конфігурації проекту через java класи (ще є спосіб конфігурації через xml файли), то з’явився ще один шар — config. Отже, серверна частина повністю написана мовою програмування

високого рівня Java, а інтерфейс користувача — на мові розмітки HTML з використанням технології JSP, що динамічно генерувати HTML сторінки.

5.3 Структура бази даних

Для розробки проекту було вирішено використовувати базу даних MySQL. Це одна з найпоширеніших на даний момент система управління базами даних [13]. MySQL має компактний багатопотоковий сервер баз даних і є гарним рішенням для малих і середніх застосувань [14].

На рисунку 5.2 зображено схему розробленої бази даних.

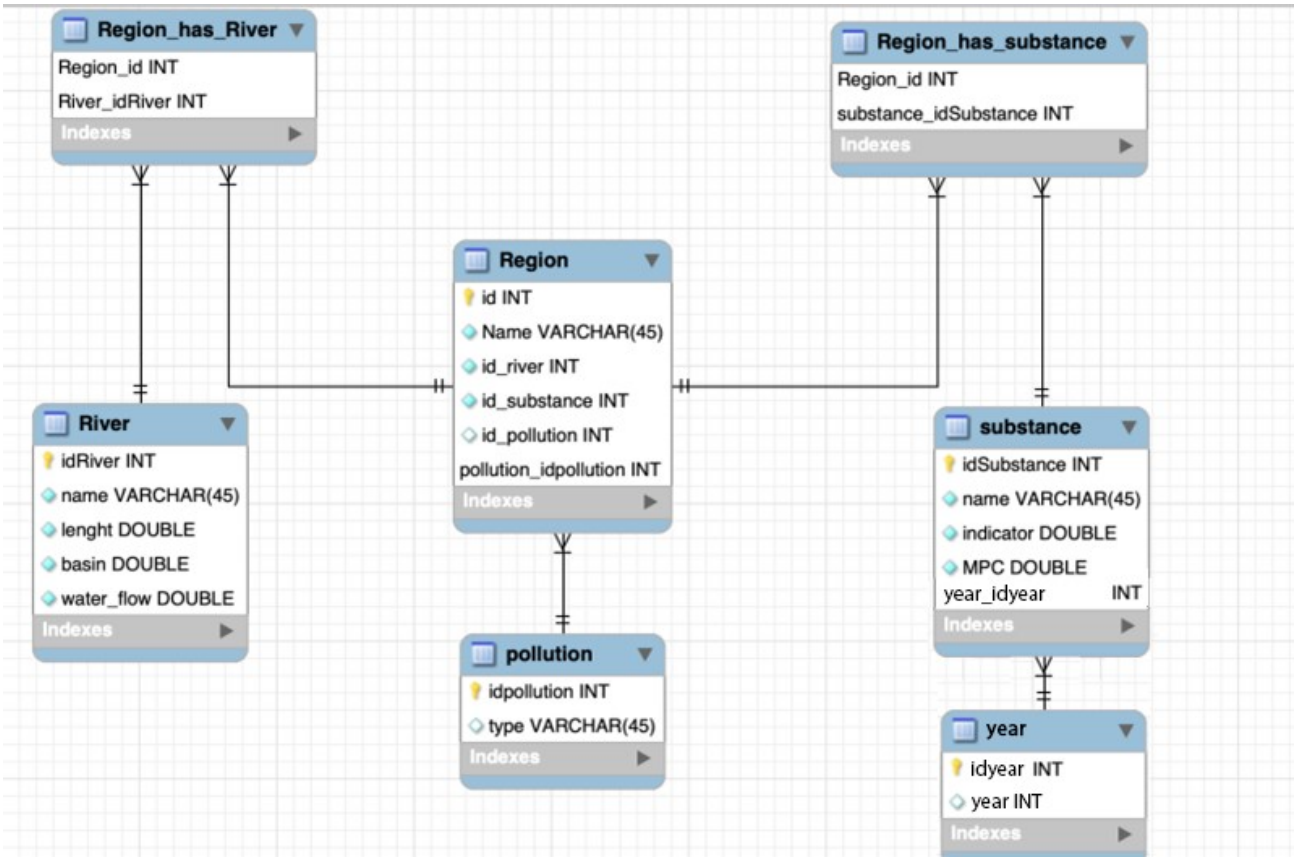

Рисунок 5.2 – Схема розробленої бази даних

Основною таблицею БД э таблиця «Регіон», яка описує область для дослідження. З нею зв’язком «Багато до багатьох» зв’язані таблиці «Річка» та «Речовина». В свою чергу таблиця «Речовина» пов’язана з таблицею «Рік», за допомогою якої можна відсортувати результати досліджень води за роками.

Також зв’язком «Один до одного» зв’язана з таблицею «Регіон» таблиця «Забрудненість», яка відповідає за збереження рівня забрудненості для кожної області [15].

5.4 Користувацький інтерфейс та можливості веб-додатку

На головному вікні веб-додатку знаходиться інтерактивна мапа, що слугує основним інтерфейсом для взаємодії користувача з іншими модулями програми та сервером, статична мапа з найбільшими річками, що складають основу водної екосистеми України. Також, було створено інтерактивну мапу з детальним зображенням річки та основними відомостями про неї, яка є основною в обраній області.

Користувачу надано можливість працювати також і з інтерактивною таблицею, яка містить інформацію про вміст у воді речовин, та їх перевищення гранично допустимих концентрацій.

Як мапа, так і дані в таблиці змінюються в залежності від взаємодії користувача з основною мапою

Зображення головного вікна програми можна бачити на рисунку 5.3.

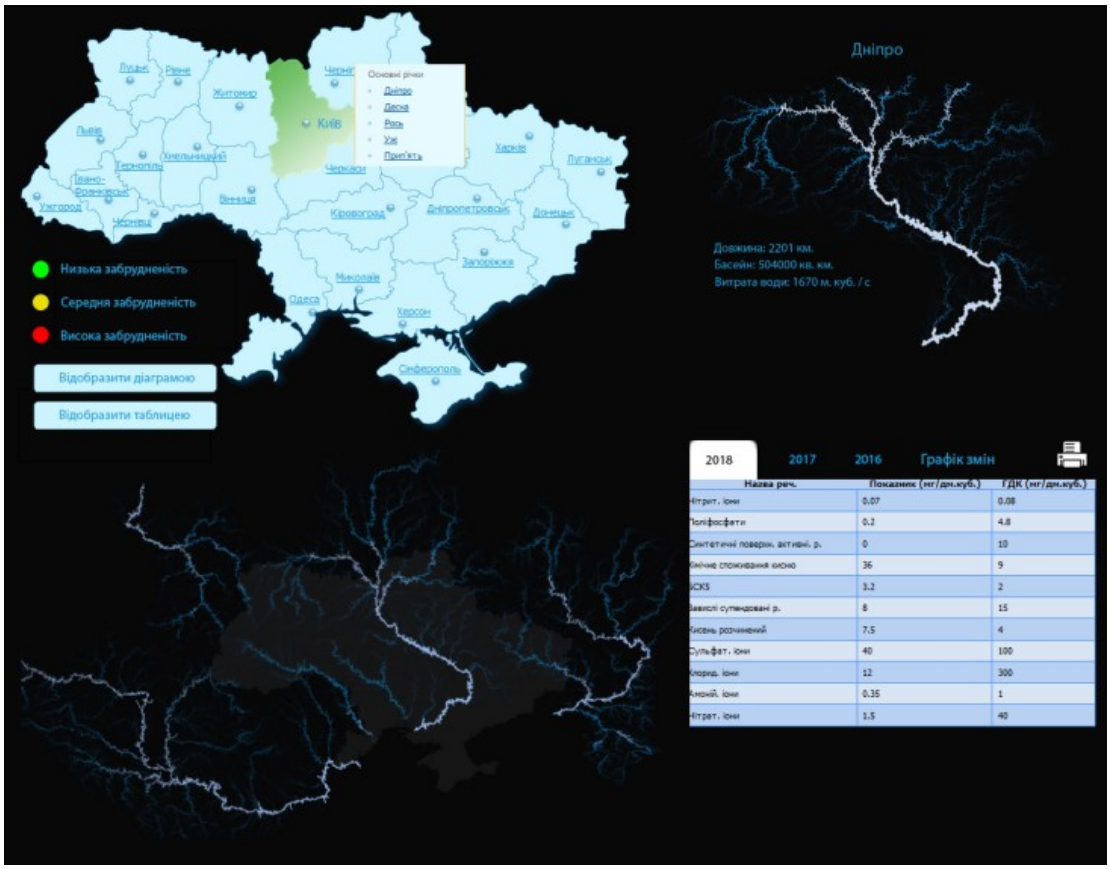

Рисунок 5.3 – Головне вікно програми

Для користування інтерактивною мапою необхідно навести курсор на будь-яку область. Після цього область буде забарвлена кольором, відповідно до стану водних ресурсів у ній. Також буде відображено список основних річок обраної області.

Після натискання на певну область, справа від інтерактивної мапи відобразиться детальна мапа основної річки обраної області з короткими відомостями про неї. Також буде оновлено дані у таблиці з речовинами відповідно до показників на обраній території. Для отримання інформації з сервера, обміну її з користувачем та іншими модулями програми, інтерактивну мапу було реалізовано мовою JavaScript та Java. Після дії користувача, відправляється запит на сервер, де він обробляється і, в залежності від потреб користувача, генерує відповідь на запит з необхідними даними. Обробка запиту відбувається мовою Java, з використанням

Spring та Hibernate фреймворків. Зокрема Hibernate фреймворк з'єднується з базою даних MySQL та забирає необхідні дані, відповідно до запиту користувача. Далі за допомогою MVC шаблону, реалізація якого в програмному продукті створена за допомогою Spring MVC фреймворка, дані передаються до користувача. Така архітектура є зручною як для програміста, адже містить чітку і зрозумілу структуру, так і для користувача, адже прискорює роботу системи [16].

Після потрапляння до клієнтської частини програми, ці дані за допомогою JavaScript надаються іншим модулям системи (рисунок 5.4).

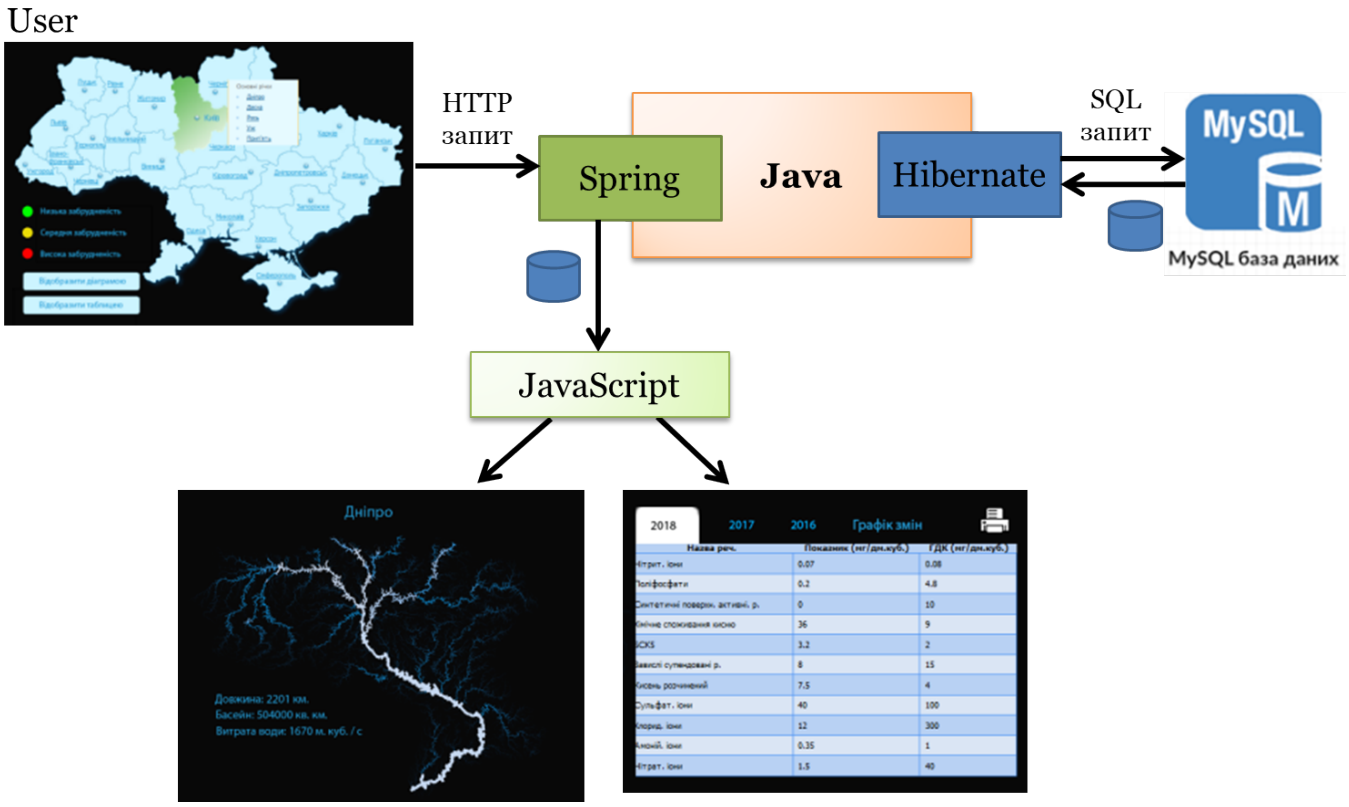

Рисунок 5.4 – Схема взаємодії інтерактивної мапи з іншими модулями

Оскільки проводити аналіз у такому вигляді мапи незручно, було додано можливість переглянути інформацію з мапи у вигляді стовпчастої та секторної діаграм (рисунок 5.5). Для відображення цих діаграм необхідно обрати функцію «Відобразити діаграмою».

Рисунок 5.5 – Діаграмне зображення інформації з мапи

Проте і в такому вигляді інформації може бути недостатньо для повного аналізу стану водних екосистем в обраній області, адже невідомо, які саме області мають незадовільний рівень забрудненості. Таким чином, використовуючи лише діаграмне відображення у користувача немає можливості отримати достатню кількість інформації.

Для того, щоб виправити цей недолік, було додано можливість переглянути дані з інтерактивної мапи у вигляді таблиці (рисунок 5.6). Для цього необхідно скористатися функцією «Відобразити таблицею». Після цього користувачеві буде показано таблицю з переліком областей навпроти кожного рівня забрудненості.

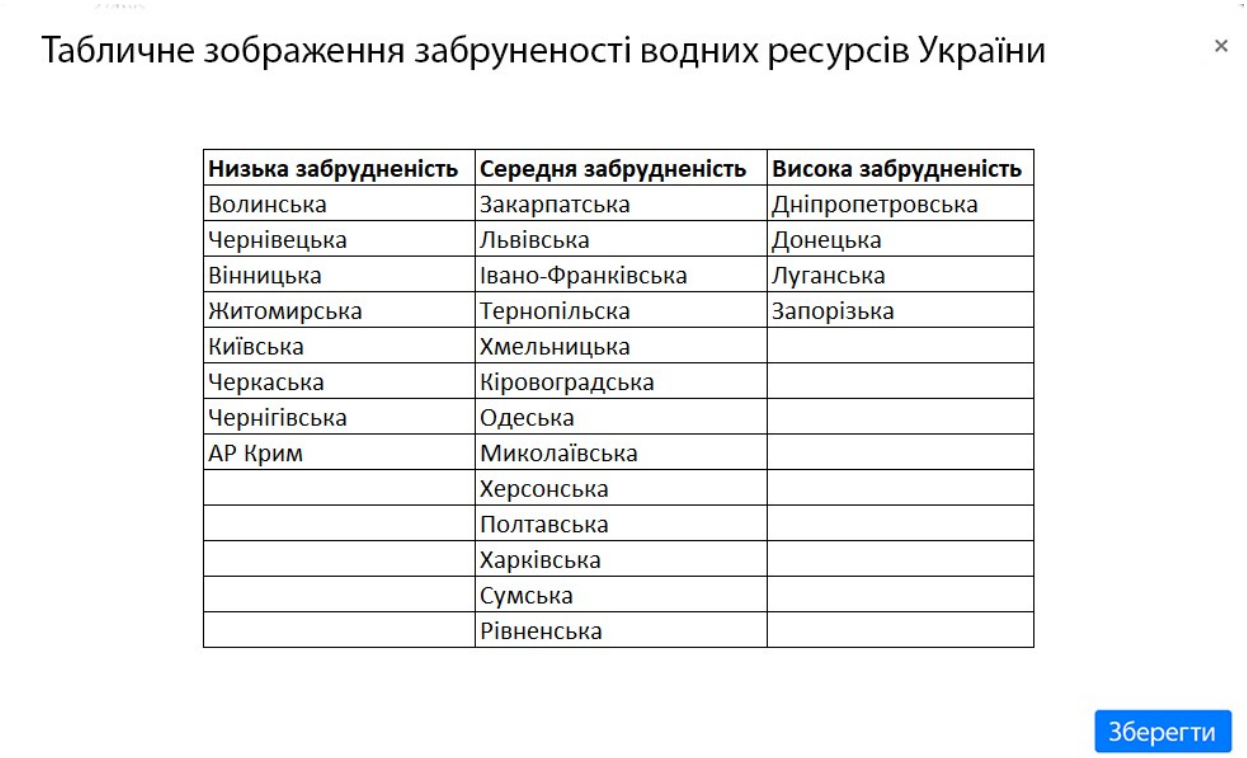

Рисунок 5.6 – Табличне зображення інформації з мапи

Для отримання даних від сервера і створення на їх основі таблиць та діаграм виконуються наступні дії:

1) Користувач обирає опцію «Відобразити таблицею» чи «Відобразити діаграмою»
2) На сервер відправляється відповідний запит
3) На сервері Java викликає Hibernate API з необхідними для отримання потрібного результату параметрами
4) Hibernate з’єднується з базою даних MySQL та отримує необхідні дані
5) Дані за допомогою Spring MVC потрапляють на клієнтську частину у вигляді відповіді від сервера
6) Дані використовуються JavaScript для побудови необхідних діаграм та таблиць

Процес отримання необхідної інформації для побудови таблиць зображено на рисунку 5.7.

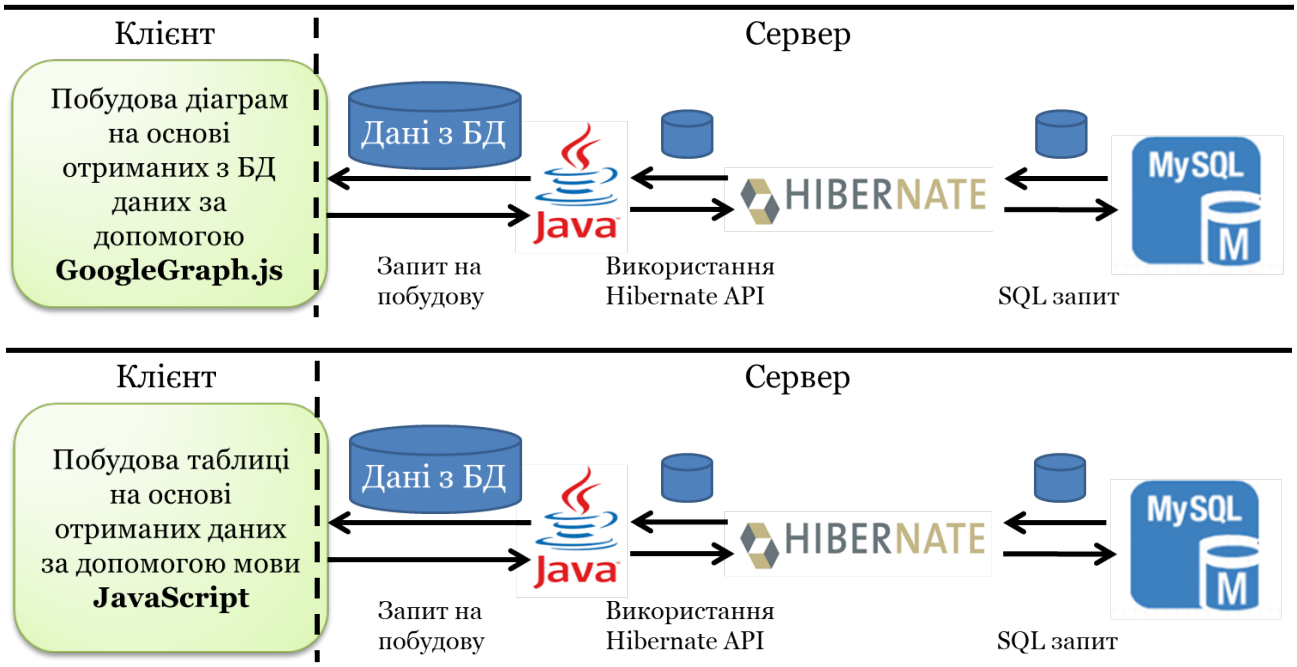

Рисунок 5.7 – Процес отримання інформації з сервера

Як дані з таблиці, так і діаграму можна зберегти в форматі Excel. Для цього достатньо обрати опцію «Зберегти». Файл буде автоматично сформовано і збережено на диск.

Перевагою розробленого підходу до збереження документів є те, що Excel файл генерується на стороні клієнта, що дає декілька переваг:

1) Зменшення інтернет-трафіка, який використовує програма. Таким чином, навіть користувачі з дуже повільним інтернет з'єднанням матимуть змогу користуватися додатком без обмежень.
2) Зменшення навантаження на сервер. Завдяки цьому сервер працюватиме, оброблятиме запити і надсилати відповіді швидше.

Побачити як саме генерується та зберігається Excel файл можна за допомогою (рисунок 5.8).

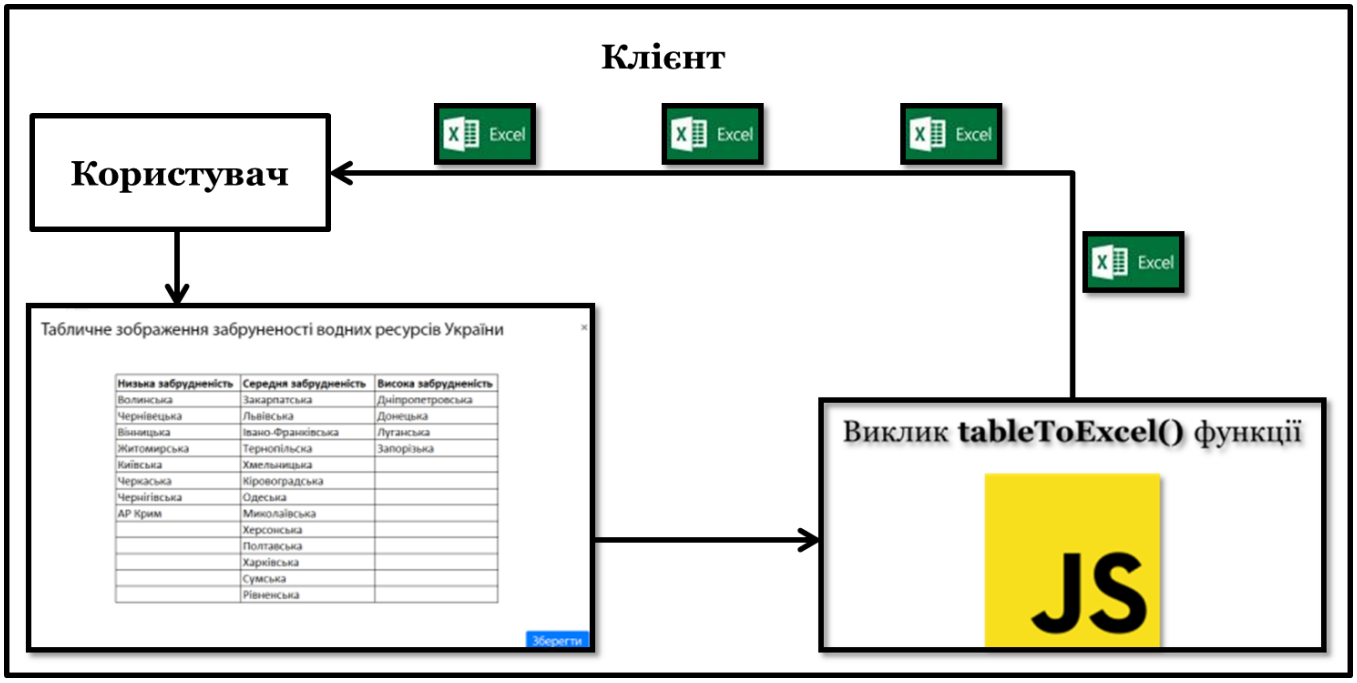

Рисунок 5.8 – Зображення процесу генерування Excel файлу

Після натискання на область інтерактивної мапи, зміниться і контент таблиці з речовинами та їх гранично допустимим значенням. Дані у цій таблиці зібрані за останні три роки, для того, щоб користувач міг побачити тенденцію розвитку чи стихання забрудненості на тій чи іншій території (рисунок 5.9).

Рисунок 5.9 – Таблиця з показником концентрації у воді речовин

Також програма має функцію побудови графіків та порівняння показників. Для цього необхідно обрати пункт меню «Графік змін».

Після цього буде відображено графік, що побудований на основі показників таблиці (рисунок 5.10).

Рисунок 5.10 – Графік з порівнянням показників за три роки

Графіки та діаграми будуються на стороні клієнта за допомогою JavaScript бібліотеки GoogleCharts.js

Наведені у таблиці дані та графік з їх порівнянням можна зберегти у форматі Excel. Зробити це можна за допомогою функції зберігання на диск. Для виклику даної функції необхідно натиснути на відповідний пункт меню.

5.5 Висновки до розділу

Було описано можливості створеного програмного продукту, взаємодію його модулів між собою та з клієнтом. Надано опис та інструкцію з користування фунціоналом веб-додатку, схеми зв’язку клієнтської та серверної частини.

Перераховано мінімальні технічні вимоги необхідні для коректної роботи програми, описано та продемонстровано структуру бази даних.

ВИСНОВКИ

У роботі було детально розглянуто програми аналоги, які надають інформацію щодо забрудненості води на тій чи іншій території України. Було проаналізовано їх переваги та недоліки, що були враховані при створенні власного програмного продукту.

Розроблено веб-додаток, що надає можливість провести аналіз стану якості водних екосистем України.

Створений веб-додаток:

а) містить інтерактивну мапу за допомогою якої користувач може отримати необхідну актуальну інформацію для аналізу стану якості води на певній території країни;

б) дозволяє користувачу переглянути детальну мапу басейну основної річки кожної області України та переглянути короткі відомості про нього;

в) має можливість відображати інформацію як у текстовому вигляді, так і у вигляді таблиць та діаграм;

г) надає користувачу статичну мапу з зображенням основних річок України;

д) містить інтерактивну порівняльну таблицю з концентраціями у воді речовин;

е) має можливість побудови графіка з порівнянням показників з інтерактивної таблиці;

ж) має можливість зберігання даних в Excel форматі для зручного перегляду в режимі офлайн.

Використання клієнтського застосунка та веб-орієнтованість забезпечили високу доступність програмного засобу та можливість запровадити єдиний інтерфейс для взаємодії з продуктом, незалежно від операційної системи, встановленої на користувацькому пристрої та його програмному забезпеченні.

ПЕРЕЛІК ПОСИЛАНЬ

1. Лаптев Ф.Ф. Анализ воды, 1955 – 144с.
2. Алекин О.А., Семенов А.Д., Скопинцев Б.А. Руководство по химическому анализу вод суши, 1973 – 270 с.
3. Шпейзер Г.М., Минеева Л.А. Руководство по химическому анализу вод, 2006 – 55 с.
4. Фаулер М. Рефакторинг. Улучшение существующего кода [Електронний ресурс] / М. Фаулер. — Пер. с англ. — СПб: Символ-Плюс, 2003. — 432 с.
5. И.Н. Блинов, В.С. Романчик Минск: издательство «Четыре четверти», 2013. — 896 с.
6. Офіційний сайт Spring Framework [Електронний ресурс] — Режим доступу: https://spring.io/ - Дата доступу: 01.06.2017
7. Собеседование по Java EE — Spring Framework [Електронний ресурс] — Режим доступу: http://javastudy.ru/interview/jee-spring-questions-answers/ - Дата доступу: 04.06.2019
8. Документація Spring Framework. 1. Introduction to Spring Framework [Електронний ресурс] — Режим доступу: https://docs.spring.io/spring/docs/3.0.0.M4/reference/html/ch01s02.html Дата доступу: 04.06.2019
9. Документація Spring Framework. 6. The Web [Електронний ресурс] — Режим доступу: https://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html Дата доступу: 04.06.2019
10. Офіційний сайт Hibernate Framework [Електронний ресурс] — Режим доступу: http://hibernate.org/orm/ Дата доступу: 04.06.2019
11. Документація Maven Framework [Електронний ресурс] — Режим доступу: http://maven.apache.org/guides/introduction/introduction-to-the-

lifecycle.html#Lifecycle_Reference Дата доступу: 04.06.2019

12. Офіційний сайт Intelij Idea [Електронний ресурс] — Режим доступу: https://www.jetbrains.com/idea/Дата доступу: 04.06.2019
13. Гольцман В. – MySQL 5.0, 2010 – 253с.
14. Полубояров В.В. – Использование MS Sql Server 2008 Analysis Services для построения хранилищ даных, 2010 – 488 с.
15. Шварц Б., Зайцев П., Ткаченко В. – MySQL. Оптимизация производительности (2-е издание), 2010 – 823с.
16. Эрик Фримен, Элизабет Робсон – Head First. Паттерны проектирования, 2018 – 657 с.

Додаток 1

ГІС аналіз стану якості водних екосистем

Специфікація

УКР.НТУУ“КПІ”.ТМ51110_19Б

Аркушів 1

2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ«КПІ ім. Ігоря Сікорського». ТМ51110_19Б 81-1	Записка	Пояснювальна записка
Компоненти		
УКР.НТУУ«КПІ». ТМ51110_19Б 12-1	Текст програмного модулю	
УКР.НТУУ«КПІ». ТМ51110_19Б 13-1	Опис програми	

Додаток 2

ГІС аналіз стану якості водних екосистем

Текст програмного модулю

УКР.НТУУ“КПІ”. ТМ51110_19Б 12-1

Аркушів 6

2019

```
public class DataDaoImpl implements DataDao {

Query query;

@PersistenceContext

private EntityManager entityManager;


@Override

public void add(Data data) {

entityManager.persist(data);

}


@Override

public Map[String, String] addToMap(Map[String, String] map, Data data) {
  mao.put(data.getName(), entityManager.get(data));
}


@Override

public void delete(long[] ids) {

Data c;

for (long id : ids) {

c = entityManager.getReference(Data.class, id);

entityManager.remove(c);

}

}



@Override

public Data findByDataid(long id) {

Query query = entityManager.createQuery("SELECT c FROM Data c WHERE c.id = :id",
Data.class);
```

```
query.setParameter("id", id);

return (Data) query.getSingleResult();

}

@Override

public List<Data> searchData(String pattern) {

Query query = entityManager.createQuery("SELECT c FROM Data c WHERE c.name LIKE
:pattern", Data.class);

query.setParameter("pattern", "%" + pattern + "%");

return (List<Data>) query.getResultList();

}

}
```

@Controller

public class UserController {

List<String> list = new ArrayList<String>();

@Autowired

private DataService dataService;

@RequestMapping(value = "/mapUpdate", method = RequestMethod.POST)

public String registration(@ModelAttribute("mapForm") Map mapForm, BindingResult bindingResult, Model model) {

userValidator.validate(userForm, bindingResult);

if (bindingResult.hasErrors()) {

return "mapUpdate";

}

dataService.addToMap(map, tableData2016);

```
dataService.addToMap(map, tableData2017);

dataService.addToMap(map, tableData2018);

dataService.addToMap(map, graphPoints);

dataService.addToMap(map, diagramData);

dataService.addToMap(map, mapTableData);

dataService.addToMap(map, riverInfo);


return "redirect:/index";
}

@Entity
@Table(name = "data")
public class Data {

public Data(){}

public Data(String name, Long id, String type, long size){
this.name = name;
This.id = id;
this.type = type;
this.size = size;
}

public User getUser() {
return user;
}

public void setUser(User user) {
this.user = user;
}

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
```

```
private Long id;

@Column(name = "name")
String name;

@Column(name = "id")
Long id;

@Column(name = "type")
String type;

@Column(name = "size")
Long size;

@ManyToOne
@JoinColumn(name = "user_id", insertable = false, updatable = false)
private User user;

public Long getId() {
return id;
}

public void setId(Long id) {
this.id = id;
}

public String getName() {
return name;
}

public void setName(String name) {
this.name = name;
}

public Long getId() {
return id;
}

public void setId(Long id) {
```

```
This.id = id;
}
public String getType() {
return type;
}
public void setType(String type) {
this.type = type;
}
public Long getSize() {
return size;
}
public void setSize(Long size) {
this.size = size;
}
}

@Service
public class DataService {
@Autowired
DataDao dataDao;

@Transactional
public void addData(Data data) {
dataDao.add(data);
}
@Transactional(readOnly=true)
public List<Data> dataList(Long id, String type) {
return dataDao.dataList(id, type);
}
```

```
@Transactional
public void deleteData(long[] ids) {
dataDao.delete(ids);
}

@Transactional(readOnly=true)
public Data findByDataid(long id) {
return dataDao.findByDataid(id);
}

@Transactional(readOnly=true)
public List<Data> searchData(String pattern) {
return dataDao.searchData(pattern);
}
}
```

Додаток 3

ГІС аналіз стану якості водних екосистем

Опис програмного модулю

УКР.НТУУ“КПІ”. ТМ51110_19Б 13-1

Аркушів 5

2019

АНОТАЦІЯ

Розроблений програмний продукт надає можливість проведення ГІС аналізу якості стану водних ресурсів України.

Користувачами даної системи можуть бути студенти вищих навчальних заходів та працівники технічних професій. В загальному програмний продукт призначений для навчальних цілях, але, завдяки гнучкості обраних технологій, може бути розширений для використання в реальних умовах праці.

ЗМІСТ

1 ВІДОМОСТІ ПРО ПРОГРАМНИЙ МОДУЛЬ

Даний програмний модуль розроблено у середовищі InellijIDEA 2019, з використанням мов Java та JavaScript, а також GoogleChart.js бібліотеки і бази даних MySQL.

Програма призначена для ГІС аналізу стану якості водних екосистем України.

1.1. Опис логічної структури

Програмний продукт було розроблено у вигляді веб-додатку.

Розроблений програмний продукт складається з двох основних частин: інтерфейсу користувача та back-end частини.

Інтерфейс користувача розроблений за допомогою мов HTML, CSS, JavaScript та складається з наступних модулів:

- Інтерактивна мапа з областями та річками України
- Детальна мапа басейну річки обраної області
- Статична мапа з річками України
- Інтерактивна таблиця з даними про вміст у воді певних речовин та можливістю побудови графіка

Серверна логіка або back-end частина розроблена мовою Java, з використанням бази даних MySQL.

1.2. Вхідні та вихідні дані

Вхідними даними є обрана користувачем область на інтерактивній мапі веб-додатку.

Вихідними даними є таблиці, діаграми, графіки та зображення, основані на даних, що були згенеровані серверною частиною програмного продукту

2 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Програмний продукт було протестовано в наступних браузерах:

- Google Chrome версії 75.0.3770.80.
- Mozilla Firefox 67.0
- Microsoft Edge 44

При тестуванні використовувався персональний комп’ютер з процесором Intel Core i7 та 16 гб оперативної пам'яті.

Мінімальні технічні засоби на яких було протестовано програмний продукт: Intel Core i3, 256мб оперативної пам’яті.